Workgroup: LAMPS Working Group
Internet-Draft:
draft-ietf-lamps-header-protection-03
Published: 22 February 2021
Intended Status: Standards Track
Expires: 26 August 2021
Authors: D.K. Gillmor
American Civil Liberties Union

B. Hoeneisen
pEp Foundation

A. Melnikov
Isode Ltd

Header Protection for S/MIME

Abstract

S/MIME version 3.1 has introduced a feasible standardized option to accomplish Header Protection. However, few implementations generate messages using this structure, and several legacy and non-legacy implementations have revealed rendering issues at the receiving side. Clearer specifications regarding message processing, particularly with respect to header sections, are needed in order to resolve these rendering issues. Some mail user agents are also sending and receiving cryptographically-protected message headers using a different structure.

In order to help implementers to correctly compose and render email messages with Header Protection, this document updates S/MIME Header Protection specifications with additional guidance on MIME format, sender and receiver processing.

Status of This Memo

This Internet-Draft is submitted in full conformance with the provisions of BCP 78 and BCP 79.

Internet-Drafts are working documents of the Internet Engineering Task Force (IETF). Note that other groups may also distribute working documents as Internet-Drafts. The list of current Internet-Drafts is at https://datatracker.ietf.org/drafts/current/.

Internet-Drafts are draft documents valid for a maximum of six months and may be updated, replaced, or obsoleted by other documents at any time. It is inappropriate to use Internet-Drafts as reference material or to cite them other than as "work in progress."

This Internet-Draft will expire on 26 August 2021.

Copyright Notice

Table of Contents

1. Introduction

Privacy and security issues regarding email Header Protection in S/MIME have been identified for some time. Most current implementations of cryptographically-protected electronic mail protect only the body of the message, which leaves significant room for attacks against otherwise-protected messages. For example, lack of header protection allows an attacker to substitute the message subject and/or author.

This document describes two different structures for how message headers can be cryptographically protected, and provides guidance for implementers of MUAs that generate and interpret such messages. It takes particular care to ensure that messages interact reasonably well with legacy MUAs.

1.1. Two Schemes of Protected Headers

Unfortunately, there are two different schemes for cryptographically-protected email headers that may be in use on the Internet today. This document addresses them both and provides guidance to implementers.

One scheme is the form specified in S/MIME 3.1 and later, which involves wrapping a message/rfc822 MIME object with a Cryptographic Envelope. This document calls this scheme "Wrapped Message", and it is documented in more detail in [RFC8551]. Experience has shown that this form does not interact well with some legacy MUAs (see Section 1.2).

Consequently, another form of header protection is produced and consumed by some MUAs, where the protected headers are placed directly on the Cryptographic Payload, without using an intervening message/* MIME object. This document calls this scheme "Injected Headers", and it is documented in more detail in [I-D.autocrypt-lamps-protected-headers].

1.2. Problems with Wrapped Messages

Several legacy MUAs have revealed rendering issues when dealing with a message with headers protected by the Wrapped Message scheme. In some cases the user sees an attachment suggesting a forwarded email message, which -- in fact -- contains the protected email message that should be rendered directly. For these cases, the user can click on the attachment to view the protected message. However, there have also been reports of email clients displaying garbled text, or sometimes nothing at all. In those cases the email clients on the receiving side are (most likely) not fully MIME-capable.

The following shortcomings have been identified to cause these issues:

* Broken or incomplete implementations

* Lack of a simple means to distinguish "forwarded message" and "wrapped message" (for the sake of Header Protection)

* Not enough guidance with respect to handling of Header Fields on both the sending and the receiving side

1.3. Motivation

Furthermore, the need (technical) Data Minimization, which includes data sparseness and hiding all technically concealable information, has grown in importance over the past several years. In addition,

backwards compatibility must be considered when it is possible to do so without compromising privacy and security.

No mechanism for Header Protection has been standardized for PGP/MIME (Pretty Good Privacy) [RFC3156] yet. PGP/MIME developers have implemented ad-hoc header-protection, and would like to see a specification that is applicable to both S/MIME and PGP/MIME.

This document describes the problem statement (Section 2), generic use cases (Section 3) and the specification for Header Protection (Section 4) with guidance on MIME format, sender and receiver processing .

[I-D.ietf-lamps-header-protection-requirements] defines the requirements that this specification is based on.

This document is in an early draft state and contains a proposal on which to base future discussions of this topic. In any case, the final mechanism is to be determined by the IETF LAMPS WG.

1.4. Other Protocols to Protect Email Headers

A range of protocols for the protection of electronic mail (email) exists, which allows one to assess the authenticity and integrity of the email headers section or selected Header Fields from the domain-level perspective, specifically DomainKeys Identified Mail (DKIM) [RFC6376], as used by Domain-based Message Authentication, Reporting, and Conformance (DMARC) [RFC7489]. These protocols provide a domain-based reputation mechanism that can be used to mitigate some forms of unsolicited email (spam). At the same time, these protocols can provide a level of cryptographic integrity and authenticity for some headers, depending on how they are used. However, integrity protection and proof of authenticity are both tied to the domain name of the sending e-mail address, not the sending address itself, so these protocols do not provide end-to-end protection, and are incapable of providing any form of confidentiality.

1.5. Requirements Language

The key words "MUST", "MUST NOT", "REQUIRED", "SHALL", "SHALL NOT", "SHOULD", "SHOULD NOT", "RECOMMENDED", "MAY", and "OPTIONAL" in this document are to be interpreted as described in [RFC2119].

1.6. Terms

The following terms are defined for the scope of this document:

*Man-in-the-middle (MITM) attack: cf. [RFC4949], which states: "A form of active wiretapping attack in which the attacker

intercepts and selectively modifies communicated data to masquerade as one or more of the entities involved in a communication association."

Note: Historically, MITM has stood for '*Man*-in-the-middle'. However, to indicate that the entity in the middle is not always a human attacker, MITM can also stand for 'Machine-in-the-middle' or 'Meddler-in-the-middle'.

*S/MIME: Secure/Multipurpose Internet Mail Extensions (cf. [RFC8551])

*PGP/MIME: MIME Security with OpenPGP (cf. [RFC3156])

*Message: An Email Message consisting of Header Fields (collectively called "the Header Section of the message") followed, optionally, by a Body; cf. [RFC5322].

Note: To avoid ambiguity, this document does not use the terms "Header" or "Headers" in isolation, but instead always uses "Header Field" to refer to the individual field and "Header Section" to refer to the entire collection; cf. [RFC5322].

*Header Field (HF): cf. [RFC5322] Header Fields are lines beginning with a field name, followed by a colon (":"), followed by a field body (value), and terminated by CRLF.

*Header Section (HS): The Header Section is a sequence of lines of characters with special syntax as defined in [RFC5322]. It is the (top) section of a Message containing the Header Fields.

*Body: The Body is simply a sequence of bytes that follows the Header Section and is separated from the Header Section by an empty line (i.e., a line with nothing preceding the CRLF); cf [RFC5322]. It is the (bottom) section of Message containing the payload of a Message. Typically, the Body consists of a (possibly multipart) MIME [RFC2045] construct.

*MIME Header Fields: Header Fields describing content of a MIME entity [RFC2045], in particular the MIME structure. Each MIME Header Field name starts with "Content-" prefix.

*MIME Header Section (part): The collection of MIME Header Fields. "MIME Header Section" refers to a Header Sections that contains only MIME Header Fields, whereas "MIME Header Section part" refers to the MIME Header Fields of a Header Section that - in addition to MIME Header Fields - also contains non-MIME Header Fields.

*Essential Header Fields (EHF): The minimum set of Header Fields an Outer Message Header Section SHOULD contain; cf. Appendix C. 1.2.5.

*Header Protection (HP): cryptographic protection of email Header Sections (or parts of it) for signatures and/or encryption

*Protection Levels (PL): The level of protection applied to a Message, e.g. 'signature and encryption' or 'signature only' (cf. Section 3.2).

*Protected: Portions of a message that have had any Protection Levels applied.

*Protected Message: A Message that has had any Protection Levels applied.

*Unprotected: Portions of a Message that has had no Protection Levels applied.

*Unprotected Message: A Message that has had no Protection Levels applied.

*Submission Entity: The entity which executes further processing of the Message (incl. transport towards the receiver), after protection measures have been applied to the Message.

Note: The Submission Entity varies among implementations, mainly depending on the stage where protection measures are applied: E.g. a Message Submission Agent (MSA) [RFC6409] or another (proprietary) solution. The latter is particularly relevant, if protection is implemented as a plugin solution. Some implementations may determine the destination recipients by reading the To, Cc and Bcc Header Fields of the Outer Message.

*Original Message (OrigM): The Message to be protected before any protection-related processing has been applied on the sending side. If the source is not a "message/rfc822" Message, OrigM is defined as the "virtual" Message that would be constructed for sending it as unprotected email.

*Inner Message (InnerM): The Message to be protected which has had wrapping and protection measures aapplied on the sending side OR the resulting Message once decryption and unwrapping on the receiving side has been performed. Typically, the Inner Message is in clear text. The Inner Message is a subset of (or the same as) the Original Message. The Inner Message must be the same on the sending and the receiving side.

*Outer Message (OuterM): The Message as provided to the Submission Entity or received from the last hop respectively. The Outer Message normally differs on the sending and the receiving side (e.g. new Header Fields are added by intermediary nodes).

*Receiving User Facing Message (RUFM): The Message used for rendering at the receiving side. Typically this is the same as the Inner Message.

*Data Minimization: Data sparseness and hiding of all technically concealable information whenever possible.

*Cryptographic Layer, Cryptographic Payload, Cryptographic Envelope, Structural Headers, and MUA are all used as defined in [I-D.dkg-lamps-e2e-mail-guidance]

*User-Facing Headers are defined in [I-D.autocrypt-lamps-protected-headers].

*Legacy MUA: a MUA that does not understand protected headers as described in this document. A Legacy Non-Crypto MUA is incapable of doing any end-to-end cryptographic operations. A Legacy Crypto MUA is capable of doing cryptographic operations, but does not understand or generate protected headers.

*Wrapped Message: The protected headers scheme that uses the mechanism described in [RFC8551], where the Cryptographic Payload is a message/rfc822 or message/global MIME object.

*Injected Headers: The protected headers scheme that uses the mechanism described in [I-D.autocrypt-lamps-protected-headers], where the protected headers are inserted on the Cryptographic Payload directly.

*Header Confidentiality Policy: documented in Section 4.1.2.2

2. Problem Statement

The LAMPS charter contains the following Work Item:

> Update the specification for the cryptographic protection of email headers -- both for signatures and encryption -- to improve the implementation situation with respect to privacy, security, usability and interoperability in cryptographically-protected electronic mail. Most current implementations of cryptographically-protected electronic mail protect only the body of the message, which leaves significant room for attacks against otherwise-protected messages.

In the following a set of challenges to be addressed:

[[TODO: Enhance this section, add more items to the following.]]

2.1. Privacy

*(Technical) Data Minimization, which includes data sparseness and hiding all technically concealable information whenever possible

2.2. Security

*Prevent MITM attacks (cf. [RFC4949])

2.3. Usability

*Improved User interaction / User experience, in particular at the receiving side

2.4. Interoperability

*Interoperability with [RFC8551] implementations

3. Use Cases

In the following, the reader can find a list of the generic use cases that need to be addressed for Messages with Header Protection (HP). These use cases apply regardless of technology (S/MIME, PGP/MIME, etc.) used to achieve HP.

3.1. Interactions

The following use cases assume that at least the sending side supports Header Protection as specified in this document. Receiving sides that support this specification are expected to be able to distinguish between Messages that use Header Protection as specified in this document, and (legacy) Mail User Agents (MUAs) which do not implement this specification.

[[TODO: Verify once solution is stable and update last sentence.]]

3.1.1. Main Use Case

Both the sending and receiving side (fully) support Header Protection as specified in this document.

The main use case is specified in Section 4.1.

3.1.2. Backward Compatibility Use Cases

Regarding backward compatibility, the main distinction is based on whether or not the receiving side conforms to MIME according to [RFC2046], ff., which in particular also includes Section 2 of

[RFC2049] on "MIME Conformance". The following excerpt is contextually relevant:

> A mail user agent that is MIME-conformant MUST:
>
> [...]
>
> -- Recognize and display at least the RFC822 message encapsulation (message/rfc822) in such a way as to preserve any recursive structure, that is, displaying or offering to display the encapsulated data in accordance with its media type.
>
> -- Treat any unrecognized subtypes as if they were "application/octet-stream".
>
> [...]
>
> An MUA that meets the above conditions is said to be MIME-conformant. A MIME-conformant MUA is assumed to be "safe" to send virtually any kind of properly-marked data to users of such mail systems, because these systems are, at a minimum, capable of treating the data as undifferentiated binary, and will not simply splash it onto the screen of unsuspecting users.

[[TODO: The compatibility of legacy HP systems with this new solution, and how to handle issues surrounding future maintenance for these legacy systems, will be decided by the LAMPS WG.]]

3.1.2.1. Receiving Side MIME-Conformant

The sending side (fully) supports Header Protection as specified in this document, while the receiving side does not support this specification. However, the receiving side is MIME-conformant according to [RFC2045], ff. (cf. Section 3.1.2).

This use case is specified in Section 4.2.1.

Note: This case should perform as expected if the sending side applies this specification as outlined in Section 4.1.

[[TODO: Verify once solution is stable and update last sentence.]]

3.1.2.2. Receiving Side Not MIME-Conformant

The sending side (fully) supports Header Protection as specified in this document, while the receiving side does not support this

specification. Furthermore, the receiving side is **not** MIME-conformant according to [RFC2045], ff. (cf. Section 3.1.2).

This use case is specified in Section 4.2.2.

3.2. Protection Levels

3.2.1. In-Scope

The following Protection Levels are in scope for this document:

a) Signature and encryption

Messages containing a cryptographic signature, which are also encrypted.

b) Signature only

Messages containing a cryptographic signature, but which are not encrypted.

3.2.2. Out-of-Scope

Legacy implementations, implementations not (fully) compliant with this document or corner-cases may lead to further Protection Levels to appear on the receiving side, such as (list not exhaustive):

- Triple wrap
- Encryption only
- Encryption before signature
- Signature and encryption, but:
  - Signature fails to validate
  - Signature validates but the signing certificate revoked
- Signature only, but:
  - with multiple valid signatures, layered atop each other

These Protection Levels, as well as any further Protection Levels not listed in Section 3.2.1 are beyond the scope of this document.

4. Specification

This section contains the specification for Header Protection in S/MIME to update and clarify Section 3.1 of [RFC8551] (S/MIME 4.0).

Note: It is likely that PGP/MIME [RFC3156] will also incorporate this specification or parts of it.

This specification applies to the Protection Levels "signature & encryption" and "signature only" (cf. Section 3.2):

Sending and receiving sides MUST implement the "signature and encryption" Protection Level, which SHOULD be used as default on the sending side.

Certain implementations may decide to send "signature only" Messages, depending on the circumstances and customer requirements. Sending sides MAY and receiving sides MUST implement "signature only" Protection Level.

It generally is NOT RECOMMENDED to send a Message with any other Protection Level. On the other hand, the receiving side must be prepared to receive Messages with other Protection Levels.

[[TODO: Further study is necessary to determine whether - and if yes to what extent - additional guidance for handling messages with other Protection Levels, e.g. "encryption only" at the receiving side should be included in this document.]]

4.1. Main Use Case

This section applies to the main use case, where the sending and receiving side (fully) support Header Protection as specified herein (cf. Section 3.1.1).

Note: The sending side specification of the main use case is also applicable to the cases where the sending side (fully) supports Header Protection as specified herein, while the receiving side does not, but is MIME-conformant according to [RFC2045], ff. (cf. Section 3.1.2 and Section 3.1.2.1).

Further backward compatibility cases are defined in Section 4.2.

4.1.1. MIME Format

4.1.1.1. Introduction

As per S/MIME version 3.1 and later (cf. [RFC8551]), the sending client MAY wrap a full MIME message in a message/RFC822 wrapper in order to apply S/MIME security services to these header fields.

To help the receiving side to distinguish between a forwarded and a wrapped message, the Content-Type header field parameter "forwarded" is added as defined in [I-D.melnikov-iana-reg-forwarded].

The simplified (cryptographic overhead not shown) MIME structure of such an Email Message looks as follows:

<Outer Message Header Section (unprotected)>

<Outer Message Body (protected)>

<MIME Header Section (wrapper)>

<Inner Message Header Section>

<Inner Message Body>

The following example demonstrates how an Original Message might be protected, i.e., the Original Message is contained as Inner Message in the Protected Body of an Outer Message. It illustrates the first Body part (of the Outer Message) as a "multipart/signed" (application/pkcs7-signature) media type:

Lines are prepended as follows:

*"O: " Outer Message Header Section

*"I: " Message Header Section

*"W: " Wrapper (MIME Header Section)

```
O: Date: Mon, 25 Sep 2017 17:31:42 +0100 (GMT Daylight Time)
O: Message-ID: <e4a483cb-1dfb-481d-903b-298c92c21f5e@m.example.net>
O: Subject: Meeting at my place
O: From: "Alexey Melnikov" <alexey.melnikov@example.net>
O: To: somebody@example.net
O: MIME-Version: 1.0
O: Content-Type: multipart/signed; charset=us-ascii; micalg=sha1;
O:  protocol="application/pkcs7-signature";
O:  boundary=boundary-AM

   This is a multipart message in MIME format.
   --boundary-AM
W: Content-Type: message/RFC822; forwarded=no
W:
I: Date: Mon, 25 Sep 2017 17:31:42 +0100 (GMT Daylight Time)
I: From: "Alexey Melnikov" <alexey.melnikov@example.net>
I: Message-ID: <e4a483cb-1dfb-481d-903b-298c92c21f5e@m.example.net>
I: MIME-Version: 1.0
I: MMHS-Primary-Precedence: 3
I: Subject: Meeting at my place
I: To: somebody@example.net
I: X-Mailer: Isode Harrier Web Server
I: Content-Type: text/plain; charset=us-ascii

   This is an important message that I don't want to be modified.

   --boundary-AM
   Content-Transfer-Encoding: base64
   Content-Type: application/pkcs7-signature

   [[base-64 encoded signature]]

   --boundary-AM--
```

The Outer Message Header Section is unprotected, while the remainder (Outer Message Body) is protected. The Outer Message Body consists of the wrapper (MIME Header Section) and the Inner Message (Header Section and Body).

The wrapper is a simple MIME Header Section with media type "message/rfc822" containing a Content-Type header field parameter "forwarded=no" followed by an empty line.

If the source is an Original (message/rfc822) Message, the Inner Message Header Section is typically the same as (or a subset of) the Original Message Header Section, and the Inner Message Body is typically the same as the Original Message Body.

The Inner Message itself may contain any MIME structure.

Note: It is still to be decided by the LAMPS WG whether or not to recommend an alternative MIME format as described in Appendix C.1.1.1 (instead of the currently standardized and above defined format).

4.1.2. Sending Side

This section describes the process an MUA should use to apply cryptographic protection to an e-mail message with header protection. We start by describing the legacy message composition process as a baseline.

4.1.2.1. Composing a Cryptographically-Protected Message Without Header Protection

[I-D.dkg-lamps-e2e-mail-guidance] describes the typical process for a legacy crypto MUA to apply cryptographic protections to an e-mail message. That guidance and terminology is replicated here for reference:

- *origbody: the traditional unprotected message body as a well-formed MIME tree (possibly just a single MIME leaf part). As a well-formed MIME tree, origbody already has structural headers (Content-*) present.

- *origheaders: the intended non-structural headers for the message, represented here as a list of (h,v) pairs, where h is a header field name and v is the associated value. Note that these are header fields that the MUA intends to be visible to the recipient of the message. In particular, if the MUA uses the Bcc header during composition, but plans to omit it from the message (see section 3.6.3 of [RFC5322]), it will not be in origheaders.

- *crypto: The series of cryptographic protections to apply (for example, "sign with the secret key corresponding to X.509 certificate X, then encrypt to X.509 certificates X and Y"). This is a routine that accepts a MIME tree as input (the Cryptographic Payload), wraps the input in the appropriate Cryptographic Envelope, and returns the resultant MIME tree as output.

The algorithm returns a MIME object that is ready to be injected into the mail system:

- *Apply crypto to origbody, yielding MIME tree output

- *For each header name and value (h,v) in origheaders:

  - -Add header h of output with value v

- *Return output

4.1.2.2. Header Confidentiality Policy

When composing an encrypted message with protected headers, the composing MUA needs a Header Confidentialiy Policy. In this document, we represent that Header Confidentiality Policy as a function hcp:

* hcp(name, val_in) --> val_out: this function takes a header field name name and initial value val_in as arguments, and returns a replacement header value val_out. If val_out is the special value null, it mean that the header in question should be omitted from the set of headers visible outside the Cryptographic Envelope.

For example, an MUA that only obscures the Subject header field by replacing it with the literal string [...] and does not offer confidentiality to any other header fields would be represented as (in pseudocode):

```
hcp(name, val_in) --> val_out: if name is 'Subject': return '[...]'
else: return val_in
```

Note that such a policy is only needed when the end-to-end protections include encryption (confidentiality). No comparable policy is needed for other end-to-end cryptographic protections (integrity and authenticity), as they are simply uniformly applied so that all header fields known by the sender have these protections.

This asymmetry is an unfortunate consequence of complexities in message delivery systems, some of which may reject, drop, or delay messages where all headers are removed from the top-level MIME object.

This document does not mandate any particular Header Confidentiality Policy, though it offers guidance for MUA implementers in selecting one in Section 4.1.3. Future documents may recommend or mandate such a policy for an MUA with specific needs. Such a recommendation might be motivated by descriptions of metadata-derived attacks, or stem from research about message deliverability, or describe new signalling mechanisms, but these topics are out of scope for this document.

4.1.2.3. Composing with "Wrapped Message" Header Protection

To compose a message using "Wrapped Message" header protection, we use those inputs described in Section 4.1.2.1 plus the Header

Confidentiality Policy hcp defined in Section 4.1.2.2. The new algorithm is:

*For header name and value (h,v) in origheaders:

-Add header h of origbody with value v

*If any of the header fields in origbody, including headers in the nested internal MIME structure, contain any 8-bit UTF-8 characters (see section section 3.7 of [RFC6532]):

-Let payload be a new MIME part with one header: Content-Type: message/global; forwarded=no, and whose body is origbody.

*Else:

-Let payload be a new MIME part with one header: Content-Type: message/rfc822; forwarded=no, and whose body is origbody.

*Apply crypto to payload, yielding MIME tree output

*If crypto contains encryption:

-Create new empty list of header field names and values newh

-For header name and value (h,v) in origheaders:

oLet newval be hcp(h, v)

oIf newval is not null:

oAppend (h,newval) to newh

-Set origheaders to newh

*For header name and value (h,v) in origheaders:

-Add header h of output with value v

*Return output

Note that the Header Confidentiality Policy hcp is ignored if crypto does not contain encryption. This is by design.

4.1.2.4. Composing with "Injected Headers" Header Protection

To compose a message using "Injected Headers" header protection, the composing MUA needs one additional input in addition to the Header Confidentiality Policy hcp defined in Section 4.1.2.2.

* legacy: a boolean value, indicating whether any recipient of the message is believed to have a legacy client. If all recipients are known to implement this draft, legacy should be set to false. (How a MUA determines the value of legacy is out of scope for this document; an initial implementation can simply set it to true)

The revised algorithm for applying cryptographic protection to a message is as follows:

* Create a new MIME leaf part legacydisplay with header Content-Type: text/plain; protected-headers="v1" and an empty body.

* if crypto contains encryption, and legacy is true:

  - For each header name and value (h,v) in origheaders:

    o If h is user-facing (see [I-D.autocrypt-lamps-protected-headers]):

      o If hcp(h,v) is not v:

        o Add h: v to the body of legacydisplay. For example, if h is Subject, and v is lunch plans?, then add the line Subject: lunch plans? to the body of legacydisplay

* If the body of legacydisplay is empty:

  - Let payload be MIME part origbody, discarding legacydisplay

* Else: (body of legacydisplay is not empty)

  - Construct a new MIME part wrapper with Content-Type: multipart/mixed

  - Give wrapper exactly two subparts: legacydisplay and origbody, in that order.

  - Let payload be MIME part wrapper

* For each header name and value (h,v) in origheaders:

  - Add header h of MIME part payload with value v

*Set the protected-headers parameter on the Content-Type of payload to v1

*Apply crypto to payload, producing MIME tree output

*If crypto contains encryption:

-Create new empty list of header field names and values newh

-For header name and value (h,v) in origheaders:

oLet newval be hcp(h, v)

oIf newval is not null:

oAdd newh[h] to newval

-Set origheaders to newh

*For each header name and value (h,v) in origheaders:

-Add header h of output with value v

*Return output

Note that both new parameters (hcp and legacy) are effectively ignored if crypto does not contain encryption. This is by design, because they are irrelevant for signed-only cryptographic protections.

4.1.2.5. Choosing Between Wrapped Message and Injected Headers

When composing a message with end-to-end cryptographic protections, an MUA SHOULD protect the headers of that message as well as the body.

An MUA MAY protect the headers of any outbound message using either the "Wrapped Message" or the "Injected Headers" style of protection. See Section 4.2 for more discussion about reasons to choose one mechanism or another.

[[TODO: this document should recommend generation of one particular scheme by default for new implementers]]

4.1.3. Default Header Confidentiality Policy

An MUA SHOULD have a sensible default Header Confidentiality Policy, and SHOULD NOT require the user to select one.

The default Header Confidentiality Policy SHOULD provide confidentiality for the Subject header field by replacing it with the literal string [...]. Most users treat the Subject of a message the same way that they treat the body, and they are surprised to find that the Subject of an encrypted message is visible.

[[TODO: select one of the two policies below the recommended default]]

4.1.3.1. Minimalist Header Confidentiality Policy

Accordingly, the most conservative recommended Header Confidentiality Policy only protects the Subject:

hcp_minimal(name, val_in) --> val_out: if name is 'Subject': return '[...]' else: return val_in

4.1.3.2. Strong Header Confidentiality Policy

Alternately, a more aggressive (and therefore more privacy-preserving) Header Confidentiality Policy only leaks a handful of fields whose absence is known to increase rates of delivery failure, and simultaneously obscures the Message-ID behind a random new one:

hcp_strong(name, val_in) --> val_out: if name in ['From', 'To', 'Cc', 'Date']: return val_in else if name is 'Subject': return '[...]' else if name is 'Message-ID': return generate_new_message_id() else: return null

The function generate_new_message_id() represents whatever process the MUA typically uses to generate a Message-ID for a new outbound message.

4.1.3.3. Offering Stronger Header Confidentiality

A MUA MAY offer even stronger confidentiality for headers of an encrypted message than described in Section 4.1.3.2. For example, it might implement an HCP that obfuscates the From field, or omits the Cc field, or ensures Date is represented in UTC (obscuring the local timezone).

The authors of this document hope that implementers with deployment experience will document their chosen Header Confidentiality Policy and the rationale behind their choice.

4.1.4. Receiving Side

An MUA that receives a cryptographically-protected e-mail will render it for the user.

The receiving MUA will render the message body, a selected subset of header fields, and (as described in [I-D.dkg-lamps-e2e-mail-guidance]) provide a summary of the cryptographic properties of the message.

Most MUAs only render a subset of header fields by default. For example, few MUAs typically render Message-Id or Received header fields for the user, but most do render From, To, Cc, Date, and Subject.

A MUA that knows how to handle a message with protected headers makes the following two changes to its behavior when rendering a message:

* If it detects that an incoming message had protected headers, it renders header fields for the message from the protected headers, ignoring the external (unprotected) headers.

* It includes information in the message's cryptographic summary to indicate the types of protection that applied to each rendered header field (if any).

A MUA that handles protected headers does *not* need to render any new header fields that it did not render before.

4.1.4.1. Identifying that a Message has Protected Headers

An incoming message can be identified as having protected headers based on one of two signals:

* The Cryptographic Payload has Content-Type: message/rfc822 or Content-Type: message/global and the parameter forwarded has a value of no. See Section 4.1.4.3 for rendering guidance.

* The Cryptographic Payload has some other Content-Type and it has parameter protected-headers set to v1. See Section 4.1.4.4 for rendering guidance.

Messages of both types exist in the wild, and a sensible MUA should be able to handle them both. They provide the same semantics and the same meaning.

4.1.4.2. Updating the Cryptographic Summary

Regardless of whether a cryptographically-protected message has protected headers, the cryptographic summary of the message should be modified to indicate what protections the headers have.

Each header individually has exactly one the following protections:

- unprotected (this is the case for all headers in messages that have no protected headers)

- signed-only (bound into the same validated signature as the enclosing message, but also visible in transit)

- encrypted-only (only appears within the cryptographic payload; the corresponding external header was either omitted or obfuscated)

- encrypted-and-signed (same as encrypted, but additionally is under a validatd signature)

Note that while the message itself may be encrypted-and-signed, some headers may be replicated on the outside of the message (e.g. Date) Those headers would be signed-only, despite the message itself being encrypted-and-signed.

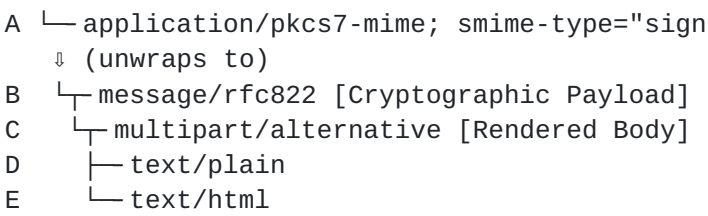
Rendering this information is likely to be complex and messy --- users may not understand it. It is beyond the scope of this document to suggest any specific graphical affordances or user experience. Future work should include examples of successful rendering of this information.

4.1.4.3. Rendering a Wrapped Message

When the Cryptographic Payload has Content-Type of message/rfc822 or message/global, and the parameter forwarded is set to no, the values of the protected headers are drawn from the headers of the Cryptographic Payload, and the body that is rendered is the body of the Cryptographic Payload.

4.1.4.3.1. Example Signed-Only Wrapped Message

Consider a message with this structure, where the MUA is able to validate the cryptographic signature:



The message body should be rendered the same way as this message:

```
C └┬╴multipart/alternative
D  ├── text/plain
E  └── text/html
```

It should render header fields taken from part C.

Its cryptographic summary should indicates that the message was signed and all rendered header fields were included in the signature.

The MUA SHOULD ignore header fields from part A for the purposes of rendering.

4.1.4.3.2. Example Encrypted-and-Signed Wrapped Message

Consider a message with this structure, where the MUA is able to validate the cryptographic signature:

```
F └── application/pkcs7-mime; smime-type="enveloped-data"
   ↧ (decrypts to)
G  └── application/pkcs7-mime; smime-type="signed-data"
    ⇩ (unwraps to)
H   └┬╴message/rfc822 [Cryptographic Payload]
I    └┬╴multipart/alternative [Rendered Body]
J     ├── text/plain
K     └── text/html
```

The message body should be rendered the same way as this message:

```
I └┬╴multipart/alternative
J  ├── text/plain
K  └── text/html
```

It should render headers taken from part I.

Its cryptographic summary should indicates that the message was signed and encrypted. Each rendered header field found in I should be compared against the header field of the same name from F. If the value found in F matches the value found in I, the header field should be marked as signed-only. If no matching header field was found in F, or the value found did not match the value from I, the header field should be marked as signed-and-encrypted.

4.1.4.4. Rendering a Message with Injected Headers

When the Cryptographic Payload does not have a Content-Type of message/rfc822 or message/global, and the parameter protected-headers is set to v1, the values of the protected headers are drawn from the headers of the Cryptographic Payload, and the body that is rendered is the Cryptographic Payload itself.

4.1.4.4.1. Example Signed-only Message with Injected Headers

```
L └─ application/pkcs7-mime; smime-type="signed-data"
   ⇩ (unwraps to)
M  └┬ multipart/alternative [Cryptographic Payload + Rendered Body]
N   ├─ text/plain
O   └─ text/html
```

The message body should be rendered the same way as this message:

```
M └┬ multipart/alternative
N  ├─ text/plain
O  └─ text/html
```

It should render header fieldss taken from part M.

Its cryptographic summary should indicates that the message was signed and all rendered header fields were included in the signature.

The MUA SHOULD ignore header fields from part L for the purposes of rendering.

4.1.4.4.2. Example Signed-and-Encrypted Message with Injected Headers

Consider a message with this structure, where the MUA is able to validate the cryptographic signature:

```
P └─ application/pkcs7-mime; smime-type="enveloped-data"
   ↧ (decrypts to)
Q  └─ application/pkcs7-mime; smime-type="signed-data"
    ⇩ (unwraps to)
R   └┬ multipart/alternative [Cryptographic Payload + Rendered Body]
S    ├─ text/plain
T    └─ text/html
```

The message body should be rendered the same way as this message:

```
R └┬ multipart/alternative
S  ├─ text/plain
T  └─ text/html
```

It should render headers taken from part R.

Its cryptographic summary should indicates that the message was signed and encrypted. As in Section 4.1.4.3.2, each rendered header field found in R should be compared against the header field of the same name from P. If the value found in P matches the value found in R, the header field should be marked as signed-only. If no matching header field was found in P, or the value found did not match the

value from R, the header field should be marked as signed-and-encrypted.

4.1.4.4.3. Do Not Render Legacy Display Part

As described [I-D.autocrypt-lamps-protected-headers], a message with cryptographic confidentiality protection MAY include a "Legacy Display" part for backward-compatibility with legacy MUAs

The receiving MUA SHOULD avoid rendering the Legacy Display part to the user at all, since it is aware of and can render the actual Protected Headers.

If a Legacy Display part is detected, it and its enclosing multipart/mixed wrapper should be discarded before rendering.

4.1.4.4.3.1. Legacy Display Detection Algorithm

A receiving MUA acting on a message SHOULD detect the presence of a Legacy Display part and the corresponding "original body" with the following simple algorithm:

* Check that all of the following are true for the message:

* The Cryptographic Envelope must contain an encrypting Cryptographic Layer

* The Cryptographic Payload must have a Content-Type of multipart/mixed

* The Cryptographic Payload must have exactly two subparts

* The first subpart of the Cryptographic Payload must have a Content-Type of text/plain or text/rfc822-headers

* The first subpart of the Cryptographic Payload's Content-Type must contain a property of protected-headers, and its value must be v1.

* If all of the above are true, then the first subpart is the Legacy Display part, and the second subpart is the "original body". Otherwise, the message does not have a Legacy Display part.

4.1.4.4.3.2. Legacy Display Example

Consider a message with this structure, where the MUA is able to validate the cryptographic signature:

```
U └─ application/pkcs7-mime; smime-type="enveloped-data"
   ↧ (decrypts to)
V  └─ application/pkcs7-mime; smime-type="signed-data"
    ⇩ (unwraps to)
W   └┬ multipart/mixed [Cryptographic Payload]
X    ├─ text/plain [Legacy Display]
Y    └┬ multipart/alternative [Rendered Body]
Z     ├─ text/plain
A'    └─ text/html
```

The message body should be rendered the same way as this message, effectively hiding the Legacy Display part (X) and its wrapper:

```
Y └┬ multipart/alternative
Z  ├─ text/plain
A' └─ text/html
```

It should render headers taken from part W, following the same guidance as in Section 4.1.4.4.2 and Section 4.1.4.3.2 about the cryptographic status of each rendered header field.

4.1.4.5. Affordances for Debugging and Troubleshooting

Note that advanced users of an MUA may need access to the original message, for example to troubleshoot problems with the MUA itself, or problems with the SMTP transport path taken by the message.

A MUA that applies these rendering guidelines SHOULD ensure that the full original source of the message as it was received remains available to such a user for debugging and troubleshooting.

4.1.4.6. Composing a Reply to an Encrypted Message with Protected Headers

When composing a reply to an encrypted message with protected headers, the MUA is acting both as a receiving MUA and as a sending MUA. Special guidance applies here, as things can go wrong in at least two ways: leaking previously-confidential information, and replying to the wrong party.

4.1.4.6.1. Avoid Leaking Encrypted Headers in Reply

As noted in [I-D.dkg-lamps-e2e-mail-guidance], an MUA in this position MUST NOT leak previously-encrypted content in the clear in a followup message. The same is true for protected headers.

Values from any header field that was identified as either encrypted or signed-and-encrypted based on the steps outlined above MUST NOT be placed in cleartext output when generating a message.

In particular, if Subject was encrypted, and it is copied into the draft encrypted reply, the replying MUA MUST obfuscate the Subject field in the cleartext header as described above.

[[TODO: formally describe how a replying MUA should generate a message-specific Header Protection policy based on the cryptographic status of the headers of the incoming message]]

4.1.4.6.2. Avoid Misdirected Replies to Encrypted Messages with Protected Headers

When replying to a message, the Composing MUA typically decides who to send the reply to based on:

* the Reply-To, Mail-Followup-To, or From headers

* optionally, the other To or Cc headers (if the user chose to "reply all")

When a message has protected headers, the replying MUA MUST populate the destination fields of the draft message using the protected headers, and ignore any unprotected headers.

This mitigates against an attack where Mallory gets a copy of an encrypted message from Alice to Bob, and then replays the message to Bob with an additional Cc to Mallory's own e-mail address in the message's outer header.

If Bob knows Mallory's certificate already, and he replies to such a message without following the guidance in this section, it's likely that his MUA will encrypt the cleartext of the message directly to Mallory.

4.1.4.7. Implicitly-rendered Header Fields

While From and To and Cc and Subject and Date are often explicitly rendered to the user, some header fields do affect message display, without being explicitly rendered.

For example, Message-Id, References, and In-Reply-To header fields may collectively be used to place a message in a "thread" or series of messages.

In another example, Section 4.1.4.6.2 observes that the value of the Reply-To field can influence the draft reply message. So while the user may never see the Reply-To header directly, it is implicitly "rendered" when the user interacts with the message by replying to it.

An MUA that depends on any implicitly-rendered header field in a message with protected headers SHOULD use the value from the protected header, and SHOULD NOT use any value found outside the cryptographic protection.

4.1.4.8. Unprotected Headers Added in Transit

Some headers are legitimately added in transit, and could not have been known to the sender at message composition time.

The most common of these headers are Received and DKIM-Signature, neither of which are typically rendered, either explicitly or implicitly.

If a receiving MUA has specific knowledge about a given header field, including that:

- the header field would not have been known to the original sender, and
- the header field might be rendered explicitly or implicitly,

then the MUA MAY decide to operate on the value of that header field from the unprotected header section, even though the message has protected headers.

The MUA MAY prefer to verify that the headers in question have additional transit-derived cryptographic protections (e.g., to test whether they are covered by a valid DKIM-Signature) before rendering or acting on them.

Specific examples appear below.

4.1.4.8.1. Mailing list headers: List-* and Archived-At

If the message arrives through a mailing list, the list manager itself may inject headers (most of which start with List-) in the message:

- List-Archive
- List-Subscribe
- List-Unsubscribe
- List-Id
- List-Help
- List-Post

*Archived-At

For some MUAs, these headers are implicitly rendered, by providing buttons for actions like "Subscribe", "View Archived Version", "Reply List", "List Info", etc.

An MUA that receives a message with protected headers that contains these header fields in the unprotected section, and that has reason to believe the message is coming through a mailing list MAY decide to render them to the user (explicitly or implicitly) even though they are not protected.

FIXME: other examples of unprotected transit headers?

4.2. Backward Compatibility Use Cases

4.2.1. Receiving Side MIME-Conformant

This section applies to the case where the sending side (fully) supports Header Protection as specified in this document, while the receiving side does not support this specification, but is MIME-conformant according to [RFC2045], ff. (cf. Section 3.1.2 and Section 3.1.2.1)

The sending side specification of the main use case (cf. Section 4.1) MUST ensure that receiving sides can still recognize and display or offer to display the encapsulated data in accordance with its media type (cf. [RFC2049], Section 2). In particular, receiving sides that do not support this specification, but are MIME-conformant according to [RFC2045], ff. can still recognize and display the Message intended for the user.

[[TODO: Verify once solution is stable and update last sentence.]]

4.2.2. Receiving Side Not MIME-Conformant

This section applies to cases where the sending side (fully) supports Header Protection as specified in this document, while the receiving side neither supports this specification **nor** is MIME-conformant according to [RFC2045], ff. (cf. Section 3.1.2 and Section 3.1.2.2).

[I-D.autocrypt-lamps-protected-headers] describes a possible way to achieve backward compatibility with existing S/MIME (and PGP/MIME) implementations that predate this specification and are not MIME-conformant (Legacy Display) either. It mainly focuses on email clients that do not render emails which utilize header protection in a user friendly manner, which may confuse the user. While this has been observed occasionally in PGP/MIME (cf. [RFC3156]), the extent of this problem with S/MIME implementations is still unclear. (Note:

At this time, none of the samples in [I-D.autocrypt-lamps-protected-headers] apply header protection as specified in Section 3.1 of [RFC8551], which is wrapping as Media Type "message/RFC822".)

Should serious backward compatibility issues with rendering at the receiving side be discovered, the Legacy Display format described in [I-D.autocrypt-lamps-protected-headers] may serve as a basis to mitigate those issues (cf. Section 4.2).

Another variant of backward compatibility has been implemented by pEp [I-D.pep-email], i.e. pEp Email Format 1.0. At this time pEp has implemented this for PGP/MIME, but not yet S/MIME.

5. Usability Considerations

This section describes concerns for MUAs that are interested in easy adoption of header protection by normal users.

While they are not protocol-level artifacts, these concerns motivate the protocol features described in this document.

See also the Usability section in [I-D.dkg-lamps-e2e-mail-guidance].

5.1. Mixed Protections Within a Message Are Hard To Understand

[[TODO]]

5.2. Users Should Not Have To Choose a Header Confidentiality Policy

[[TODO]]

6. Security Considerations

[[TODO]]

7. Privacy Considerations

[[TODO]]

8. IANA Considerations

This document requests no action from IANA.

[[RFC Editor: This section may be removed before publication.]]

9. Acknowledgments

The authors would like to thank the following people who have provided helpful comments and suggestions for this document: Berna Alp, Claudio Luck, David Wilson, Hernani Marques, juga, Krista

Bennett, Kelly Bristol, Lars Rohwedder, Robert Williams, Russ Housley, Sofia Balicka, Steve Kille, Volker Birk, and Wei Chuang.

10. References

10.1. Normative References

[I-D.dkg-lamps-e2e-mail-guidance]
Gillmor, D. K., "Guidance on End-to-End E-mail Security", Work in Progress, Internet-Draft, draft-dkg-lamps-e2e-mail-guidance-01, 22 February 2021, <https://www.ietf.org/archive/id/draft-dkg-lamps-e2e-mail-guidance-01.txt>.

[I-D.ietf-lamps-header-protection-requirements]
Melnikov, A. and B. Hoeneisen, "Problem Statement and Requirements for Header Protection", Work in Progress, Internet-Draft, draft-ietf-lamps-header-protection-requirements-01, 29 October 2019, <https://www.ietf.org/archive/id/draft-ietf-lamps-header-protection-requirements-01.txt>.

[RFC2045] Freed, N. and N. Borenstein, "Multipurpose Internet Mail Extensions (MIME) Part One: Format of Internet Message Bodies", RFC 2045, DOI 10.17487/RFC2045, November 1996, <https://www.rfc-editor.org/info/rfc2045>.

[RFC2046] Freed, N. and N. Borenstein, "Multipurpose Internet Mail Extensions (MIME) Part Two: Media Types", RFC 2046, DOI 10.17487/RFC2046, November 1996, <https://www.rfc-editor.org/info/rfc2046>.

[RFC2049] Freed, N. and N. Borenstein, "Multipurpose Internet Mail Extensions (MIME) Part Five: Conformance Criteria and Examples", RFC 2049, DOI 10.17487/RFC2049, November 1996, <https://www.rfc-editor.org/info/rfc2049>.

[RFC2119] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", BCP 14, RFC 2119, DOI 10.17487/RFC2119, March 1997, <https://www.rfc-editor.org/info/rfc2119>.

[RFC5322] Resnick, P., Ed., "Internet Message Format", RFC 5322, DOI 10.17487/RFC5322, October 2008, <https://www.rfc-editor.org/info/rfc5322>.

[RFC8551] Schaad, J., Ramsdell, B., and S. Turner, "Secure/Multipurpose Internet Mail Extensions (S/MIME) Version 4.0 Message Specification", RFC 8551, DOI 10.17487/

RFC8551, April 2019, <https://www.rfc-editor.org/info/rfc8551>.

10.2. Informative References

[I-D.autocrypt-lamps-protected-headers]
Einarsson, B. R., juga, and D. K. Gillmor, "Protected Headers for Cryptographic E-mail", Work in Progress, Internet-Draft, draft-autocrypt-lamps-protected-headers-02, 20 December 2019, <https://www.ietf.org/archive/id/draft-autocrypt-lamps-protected-headers-02.txt>.

[I-D.dkg-lamps-samples] Gillmor, D. K., "S/MIME Example Keys and Certificates", Work in Progress, Internet-Draft, draft-dkg-lamps-samples-05, 18 February 2021, <https://www.ietf.org/archive/id/draft-dkg-lamps-samples-05.txt>.

[I-D.melnikov-iana-reg-forwarded]
Melnikov, A. and B. Hoeneisen, "IANA Registration of Content-Type Header Field Parameter 'forwarded'", Work in Progress, Internet-Draft, draft-melnikov-iana-reg-forwarded-00, 4 November 2019, <https://www.ietf.org/archive/id/draft-melnikov-iana-reg-forwarded-00.txt>.

[I-D.pep-email] Marques, H., "pretty Easy privacy (pEp): Email Formats and Protocols", Work in Progress, Internet-Draft, draft-pep-email-01, 2 November 2020, <https://www.ietf.org/archive/id/draft-pep-email-01.txt>.

[pEp.mixnet] pEp Foundation, "Mixnet", June 2020, <https://dev.pep.foundation/Mixnet>.

[RFC3156] Elkins, M., Del Torto, D., Levien, R., and T. Roessler, "MIME Security with OpenPGP", RFC 3156, DOI 10.17487/RFC3156, August 2001, <https://www.rfc-editor.org/info/rfc3156>.

[RFC4949] Shirey, R., "Internet Security Glossary, Version 2", FYI 36, RFC 4949, DOI 10.17487/RFC4949, August 2007, <https://www.rfc-editor.org/info/rfc4949>.

[RFC6376] Crocker, D., Ed., Hansen, T., Ed., and M. Kucherawy, Ed., "DomainKeys Identified Mail (DKIM) Signatures", STD 76, RFC 6376, DOI 10.17487/RFC6376, September 2011, <https://www.rfc-editor.org/info/rfc6376>.

[RFC6409] Gellens, R. and J. Klensin, "Message Submission for Mail", STD 72, RFC 6409, DOI 10.17487/RFC6409, November 2011, <https://www.rfc-editor.org/info/rfc6409>.

[RFC6532] Yang, A., Steele, S., and N. Freed, "Internationalized Email Headers", RFC 6532, DOI 10.17487/RFC6532, February 2012, <https://www.rfc-editor.org/info/rfc6532>.

[RFC7489] Kucherawy, M., Ed. and E. Zwicky, Ed., "Domain-based Message Authentication, Reporting, and Conformance (DMARC)", RFC 7489, DOI 10.17487/RFC7489, March 2015, <https://www.rfc-editor.org/info/rfc7489>.

Appendix A. Test Vectors

This section contains sample messages using the different schemes described in this document. Each sample contains a MIME object, and examples of how an MUA might render it.

The cryptographic protections used in this document use the S/MIME standard, and keying material and certificates come from [I-D.dkg-lamps-samples].

For the signed-and-encrypted messages, only the Subject header is obscured.

A.1. Wrapped Message examples

The examples in this subsection use the "Wrapped Message" header protection scheme.

A.1.1. Wrapped Message: signed-only, with PKCS7 signedData

[[TODO]]

A.1.2. Wrapped Message: signed-only, using multipart/signed

[[TODO]]

A.1.3. Wrapped Message: signed-and-encrypted

[[TODO]]

A.2. Injected Headers examples

The examples in this subsection use the "Injected Headers" header protection scheme.

A.2.1. Injected Headers: signed-only, with PKCS7 signedData

[[TODO]]

A.2.2. Injected Headers: signed-only, using multipart/signed

[[TODO]]

A.2.3. Injected Headers: signed-and-encrypted with Legacy Display part

[[TODO]]

A.2.4. Injected Headers: signed-and-encrypted without Legacy Display part

[[TODO]]

A.3. Messages without Header Protection

The examples in this subsection have cryptographic protection, but no header protection. They are provided in this document as a counterexample. An MUA implementer can use these messages to verify that the reported cryptographic summary of the message indicates no header protection.

A.3.1. Unprotected Headers: signed-only, with PKCS7 signedData

[[TODO]]

A.3.2. Unprotected Headers: signed-only, using multipart/signed

[[TODO]]

A.3.3. Unprotected Headers: signed-and-encrypted

[[TODO]]

Appendix B. Additional information

B.1. Stored Variants of Messages with Bcc

Messages containing at least one recipient address in the Bcc header field may appear in up to three different variants:

1. The Message for the recipient addresses listed in To or Cc header fields, which must not include the Bcc header field neither for signature calculation nor for encryption.

2. The Message(s) sent to the recipient addresses in the Bcc header field, which depends on the implementation:

   a) One Message for each recipient in the Bcc header field separately, with a Bcc header field containing only the address of the recipient it is sent to.

b) The same Message for each recipient in the Bcc header field with a Bcc header field containing an indication such as "Undisclosed recipients", but no addresses.

c) The same Message for each recipient in the Bcc header field which does not include a Bcc header field (this Message is identical to 1. / cf. above).

3. The Message stored in the 'Sent'-Folder of the sender, which usually contains the Bcc unchanged from the original Message, i.e., with all recipient addresses.

The most privacy preserving method of the alternatives (2a, 2b, and 2c) is to standardize 2a, as in the other cases (2b and 2c), information about hidden recipients is revealed via keys. In any case, the Message has to be cloned and adjusted depending on the recipient.

Appendix C. Text Moved from Above

Note: Per an explicit request by the chair of the LAMPS WG to only present one option for the specification, the following text has been stripped from the main body of the draft. It is preserved in an Appendix for the time being and may be moved back to the main body or deleted, depending on the decision of the LAMPS WG.

C.1. MIME Format

Currently there are two options in discussion:

1. The option according to the current S/MIME specification (cf. [RFC8551])

2. An alternative option that is based on the former "memory hole" approach (cf. [I-D.autocrypt-lamps-protected-headers])

C.1.1. S/MIME Specification

Note: This is currently described in the main part of this document.

C.1.1.1. Alternative Option Autocrypt "Protected Headers" (Ex-"Memory Hole")

An alternative option (based on the former autocrypt "Memory Hole" approach) to be considered, is described in [I-D.autocrypt-lamps-protected-headers].

Unlike the option described in Appendix C.1.1, this option does not use a "message/RFC822" wrapper to unambiguously delimit the Inner Message.

Before choosing this option, the following two issues must be assessed to ensure no interoperability issues result from it:

1. How current MIME parser implementations treat non-MIME Header Fields, which are not part of the outermost MIME entity and not part of a Message wrapped into a MIME entity of media type "message/rfc822", and how such Messages are rendered to the user.

   [I-D.autocrypt-lamps-protected-headers] provides some examples for testing this.

2. MIME-conformance, i.e. whether or not this option is (fully) MIME-conformant [RFC2045] ff., in particular also Section 5.1. of [RFC2046] on "Multipart Media Type). In the following an excerpt of paragraphs that may be relevant in this context:

   The only header fields that have defined meaning for body parts are those the names of which begin with "Content-". All other header fields may be ignored in body parts. Although they should generally be retained if at all possible, they may be discarded by gateways if necessary. Such other fields are permitted to appear in body parts but must not be depended on. "X-" fields may be created for experimental or private purposes, with the recognition that the information they contain may be lost at some gateways.

   NOTE: The distinction between an RFC 822 Message and a body part is subtle, but important. A gateway between Internet and X.400 mail, for example, must be able to tell the difference between a body part that contains an image and a body part that contains an encapsulated Message, the body of which is a JPEG image. In order to represent the latter, the body part must have "Content-Type: message/rfc822", and its body (after the blank line) must be the encapsulated Message, with its own "Content-Type: image/jpeg" header field. The use of similar syntax facilitates the conversion of Messages to body parts, and vice versa, but the distinction between the two must be understood by implementors. (For the special case in which parts actually are Messages, a "digest" subtype is also defined.)

The MIME structure of an Email Message looks as follows:

<Outer Message Header Section (unprotected)>

<Outer Message Body (protected)>

  <Inner Message Header Section>

  <Inner Message Body>

The following example demonstrates how an Original Message might be protected, i.e., the Original Message is contained as Inner Message in the Protected Body of an Outer Message. It illustrates the first Body part (of the Outer Message) as a "multipart/signed" (application/pkcs7-signature) media type:

Lines are prepended as follows:

  *"O: " Outer Message Header Section

  *"I: " Message Header Section

```
O: Date: Mon, 25 Sep 2017 17:31:42 +0100 (GMT Daylight Time)
O: Message-ID: <e4a483cb-1dfb-481d-903b-298c92c21f5e@m.example.net>
O: Subject: Meeting at my place
O: From: "Alexey Melnikov" <alexey.melnikov@example.net>
O: MIME-Version: 1.0
O: Content-Type: multipart/signed; charset=us-ascii; micalg=sha1;
O:  protocol="application/pkcs7-signature";
O:  boundary=boundary-AM

   This is a multipart message in MIME format.
   --boundary-AM
I: Date: Mon, 25 Sep 2017 17:31:42 +0100 (GMT Daylight Time)
I: From: "Alexey Melnikov" <alexey.melnikov@example.net>
I: Message-ID: <e4a483cb-1dfb-481d-903b-298c92c21f5e@m.example.net>
I: MIME-Version: 1.0
I: MMHS-Primary-Precedence: 3
I: Subject: Meeting at my place
I: To: somebody@example.net
I: X-Mailer: Isode Harrier Web Server
I: Content-Type: text/plain; charset=us-ascii

   This is an important message that I don't want to be modified.

   --boundary-AM
   Content-Transfer-Encoding: base64
   Content-Type: application/pkcs7-signature

   [[base-64 encoded signature]]

   --boundary-AM--
```

The Outer Message Header Section is unprotected, while the remainder (Outer Message Body) is protected. The Outer Message Body consists of the Inner Message (Header Section and Body).

The Inner Message Header Section is the same as (or a subset of) the Original Message Header Section.

The Inner Message Body is the same as the Original Message Body.

The Original Message itself may contain any MIME structure.

C.1.2. Sending Side

To ease explanation, the following describes the case where an Original (message/rfc822) Message to be protected is present. If this is not the case, Original Message means the (virtual) Message that would be constructed for sending it as unprotected email.

C.1.2.1. Inner Message Header Fields

It is RECOMMENDED that the Inner Message contains all Header Fields of the Original Message with the exception of the following Header Field, which MUST NOT be included within the Inner Message nor within any other protected part of the Message:

- *Bcc

[[TODO: Bcc handling needs to be further specified (see also Appendix B.1). Certain MUAs cannot properly decrypt Messages with Bcc recipients.]]

C.1.2.2. Wrapper

The wrapper is a simple MIME Header Section followed by an empty line preceding the Inner Message (inside the Outer Message Body). The media type of the wrapper MUST be "message/RFC822" and MUST contain the Content-Type header field parameter "forwarded=no" as defined in [I-D.melnikov-iana-reg-forwarded]. The wrapper unambiguously delimits the Inner Message from the rest of the Message.

C.1.2.3. Cryptographic Layers / Envelope

[[TODO: Basically refer to S/MIME standards]]

C.1.2.4. Sending Side Message Processing

For a protected Message the following steps are applied before a Message is handed over to the Submission Entity:

C.1.2.4.1. Step 1: Decide on Protection Level and Information Disclosure

The implementation which applies protection to a Message must decide:

- *Which Protection Level (signature and/or encryption) shall be applied to the Message? This depends on user request and/or local policy as well as availability of cryptographic keys.

- *Which Header Fields of the Original Message shall be part of the Outer Message Header Section? This typically depends on local policy. By default, the Essential Header Fields are part of the Outer Message Header Section; cf. Appendix C.1.2.5.

- *Which of these Header Fields are to be obfuscated? This depends on local policy and/or specific Privacy requirements of the user.

By default only the Subject Header Field is obfuscated; cf. Appendix C.1.2.5.

C.1.2.4.2. Step 2: Compose the Outer Message Header Section

Depending on the decision in Appendix C.1.2.4.1, the implementation shall compose the Outer Message Header Section. (Note that this also includes the necessary MIME Header Section part for the following protection layer.)

Outer Header Fields that are not obfuscated should contain the same values as in the Original Message (except for MIME Header Section part, which depends on the Protection Level selected in Appendix C.1.2.4.1).

C.1.2.4.3. Step 3: Apply Protection to the Original Message

Depending on the Protection Level selected in Appendix C.1.2.4.1, the implementation applies signature and/or encryption to the Original Message, including the wrapper (as per [RFC8551]), and sets the resulting package as the Outer Message Body.

The resulting (Outer) Message is then typically handed over to the Submission Entity.

[[TODO: Example]]

C.1.2.5. Outer Message Header Fields

C.1.2.5.1. Encrypted Messages

To maximize Privacy, it is strongly RECOMMENDED to follow the principle of Data Minimization (cf. Section 2.1).

However, the Outer Message Header Section SHOULD contain the Essential Header Fields and, in addition, MUST contain the Header Fields of the MIME Header Section part to describe Cryptographic Layer of the protected MIME subtree as per [RFC8551].

The following Header Fields are defined as the Essential Header Fields:

*From

*To (if present in the Original Message)

*Cc (if present in the Original Message)

*Bcc (if present in the Original Message, see also Appendix B.1)

*Date

*Message-ID

*Subject

Further processing by the Submission Entity normally depends on part of these Header Fields, e.g. From and Date HFs are required by [RFC5322]. Furthermore, not including certain Header Fields may trigger spam detection to flag the Message, and/or lead to user experience (UX) issues.

For further Data Minimization, the value of the Subject Header Field SHOULD be obfuscated as follows:

* Subject: [...]

and it is RECOMMENDED to replace the Message-ID by a new randomly generated Message-ID.

In addition, the value of other Essential Header Fields MAY be obfuscated.

Non-Essential Header Fields SHOULD be omitted from the Outer Message Header Section where possible. If Non-essential Header Fields are included in the Outer Message Header Section, those MAY be obfuscated too.

Header Fields that are not obfuscated should contain the same values as in the Original Message.

If an implementation obfuscates the From, To, and/or Cc Header Fields, it may need to provide access to the clear text content of these Header Fields to the Submission Entity for processing purposes. This is particularly relevant, if proprietary Submission Entities are used. Obfuscation of Header Fields may adversely impact spam filtering.

(A use case for obfuscation of all Outer Message Header Fields is routing email through the use of onion routing or mix networks, e.g. [pEp.mixnet].)

The MIME Header Section part is the collection of MIME Header Fields describing the following MIME structure as defined in [RFC2045]. A MIME Header Section part typically includes the following Header Fields:

*Content-Type

*Content-Transfer-Encoding

*Content-Disposition

The following example shows the MIME Header Section part of an S/MIME signed Message (using application/pkcs7-mime with SignedData):

```
MIME-Version: 1.0
Content-Type: application/pkcs7-mime; smime-type=signed-data;
   name=smime.p7m
Content-Transfer-Encoding: base64
Content-Disposition: attachment; filename=smime.p7m
```

Depending on the scenario, further Header Fields MAY be exposed in the Outer Message Header Section, which is NOT RECOMMENDED unless justified. Such Header Fields may include e.g.:

*References

*Reply-To

*In-Reply-To

C.1.2.5.2. Unencrypted Messages

The Outer Message Header Section of unencrypted Messages SHOULD contain at least the Essential Header Fields and, in addition, MUST contain the Header Fields of the MIME Header Section part to describe Cryptographic Layer of the protected MIME subtree as per [RFC8551]. It may contain further Header Fields, in particular those also present in the Inner Message Header Section.

Appendix D. Document Considerations

[[RFC Editor: This section is to be removed before publication]]

This draft is built from markdown source, and its development is tracked in a git repository.

While minor editorial suggestions and nit-picks can be made as merge requests, please direct all substantive discussion to the LAMPS mailing list at spasm@ietf.org.

Appendix E. Document Changelog

[[RFC Editor: This section is to be removed before publication]]

*draft-ietf-lamps-header-protection-03

-dkg takes over from Bernie as primary author

-Add Usability section

-describe two distinct formats "Wrapped Message" and "Injected Headers"

-Introduce Header Confidentiality Policy model

-Overhaul message composition guidance

-Simplify document creation workflow, move public face to gitlab

*draft-ietf-lamps-header-protection-02

-editorial changes / improve language

*draft-ietf-lamps-header-protection-01

-Add DKG as co-author

-Partial Rewrite of Abstract and Introduction [HB/AM/DKG]

-Adding definiations for Cryptographic Layer, Cryptographic Payload, and Cryptographic Envelope (reference to [I-D.dkg-lamps-e2e-mail-guidance]) [DKG]

-Enhanced MITM Definition to include Machine- / Meddler-in-the-middle [HB]

-Relaxed definition of Original message, which may not be of type "message/rfc822" [HB]

-Move "memory hole" option to the Appendix (on request by Chair to only maintain one option in the specification) [HB]

-Updated Scope of Protection Levels according to WG discussion during IETF-108 [HB]

-Obfuscation recommendation only for Subject and Message-Id and distinguish between Encrypted and Unencrypted Messages [HB]

-Removed (commented out) Header Field Flow Figure (it appeared to be confusing as is was) [HB]

*draft-ietf-lamps-header-protection-00

-Initial version (text partially taken over from [I-D.ietf-lamps-header-protection-requirements]

Appendix F. Open Issues

[[RFC Editor: This section should be empty and is to be removed before publication.]]

- *Ensure "protected header" (Ex-Memory-Hole) option is (fully) compliant with the MIME standard, in particular also [RFC2046], Section 5.1. (Multipart Media Type) Appendix C.1.1.1.

- *Test Vectors! This should be a new appendix section, to avoid injecting large blobs of unreadable data in the main text. Once present, we can point to the relevant test vector in the main text by reference.

- *Should Outer Message Header Section (as received) be preserved for the user? (Section 4.1.4.5)

- *Decide on whether or not merge requirements from [I-D.ietf-lamps-header-protection-requirements] into this document.

- *Decide what parts of [I-D.autocrypt-lamps-protected-headers] to merge into this document.

- *Enhance Introduction Section 1 and Problem Statement (Section 2).

- *Decide on whether or not specification for more legacy HP requirements should be added to this document (Section 3.1.2).

- *Verify simple backward compatibility case (Receiving Side MIME-Conformant) is working; once solution is stable and update paragraphs in Section 4.1, Section 3.1.2.1 and Section 4.2.1 accordingly.

- *Verify ability to distinguish between Messages with Header Protection as specified in this document and legacy clients and update Section 3.1 accordingly.

- *Improve definitions of Protection Levels and enhance list of Protection Levels (Section 3.2, Section 4).

- *Privacy Considerations Section 7

- *Security Considerations Section 6

Authors' Addresses

Daniel Kahn Gillmor
American Civil Liberties Union
125 Broad St.
New York, NY, 10004

United States of America

Email: dkg@fifthhorseman.net

Bernie Hoeneisen
pEp Foundation
Oberer Graben 4
CH- CH-8400 Winterthur
Switzerland

Email: bernie.hoeneisen@pep.foundation
URI: https://pep.foundation/

Alexey Melnikov
Isode Ltd
14 Castle Mews
Hampton, Middlesex
TW12 2NP
United Kingdom

Email: alexey.melnikov@isode.com